Fragments of Bag Relational Algebra: Expressiveness and Certain Answers

Fragments of Bag Relational Algebra: Expressiveness and Certain Answers

Marco Console
School of Informatics, University of Edinburgh, United Kingdom
console.marco@gmail.com

Paolo Guagliardo
School of Informatics, University of Edinburgh, United Kingdom
paolo.guagliardo@ed.ac.uk

Leonid Libkin
School of Informatics, University of Edinburgh, United Kingdom
libkin@inf.ed.ac.uk

Abstract

While all relational database systems are based on the bag data model, much of theoretical research still views relations as sets. Recent attempts to provide theoretical foundations for modern data management problems under the bag semantics concentrated on applications that need to deal with incomplete relations, i.e., relations populated by constants and nulls. Our goal is to provide a complete characterization of the complexity of query answering over such relations in fragments of bag relational algebra.

The main challenges we face are twofold. First, bag relational algebra has more operations than its set analog (e.g., additive union, max-union, min-intersection, duplicate elimination) and the relationship between various fragments is not fully known. Thus we first fill this gap. Second, we look at query answering over incomplete data which again is more complex than in the set case: rather than certainty and possibility of answers, we now have numerical information about occurrences of tuples. We then fully classify the complexity of finding this information in all the fragments of bag relational algebra.

2012 ACM Subject Classification Theory of computation → Database theory; Theory of computation → Database query languages (principles); Theory of computation → Incomplete, inconsistent, and uncertain databases; Information systems → Relational database query languages; Information systems → Structured Query Language

Keywords and phrases bag semantics, relational algebra, expressivity, certain answers, complexity

Digital Object Identifier 10.4230/LIPIcs.ICDT.2019.5

1 Introduction

While all relational database management systems (DBMSs) use bags as the basis of their data model, much of relational database theory uses a model based on sets, thus disallowing repetitions of tuples. The presence of duplicates in real-life databases is a very important consideration that is reflected in practically all aspects of data management, such as querying, storing, and accessing data [15, 30]. Theoretical research has raised this issue several times. By the early 1990s there was agreement on what the standard collection of bag relational algebra operations is [4], and in the mid 1990s their expressiveness and complexity were thoroughly studied [18, 26], albeit in the context of the model of nested relations, or complex objects, which was the research focus back then [9, 10]. Around the same time it was noticed that the well developed theory of query optimization, especially for conjunctive queries, does not apply to bag semantics [11], and despite many attempts and partial results [23, 12], the key problem of the decidability of such optimizations remains unsolved [24]. Other languages,

in particular those with aggregates and fixpoints in the spirit of Datalog, have been studied under bag semantics as well [7, 13, 26].

More recently, bag semantics has been considered in modern data management applications that combine traditional databases and reasoning tasks. In [20], fundamental problems of data integration and data exchange are studied under bag semantics and are shown to differ rather drastically from their set semantics counterparts. In [28], a similar program is carried out for ontology based data access (OBDA), where an ontology supplements information provided by a relational database in which duplicates are allowed. What is common to these applications is that in both of them one needs to query incomplete data, that is, databases with null values. The standard approach to querying such databases, which is used in data integration, data exchange, and OBDA applications, is based on the classical notion of certain answers [22].

However, when it comes to bags, the notion of certain answers becomes more complex than under set semantics. In general, an incomplete database D represents a collection $[\![D]\!] = \{D_1, D_2, \ldots\}$ of complete databases, obtained by interpreting incomplete data in D. A tuple $\bar{a}$ is a certain answer to a query q if it is in $q(D')$ for every $D' \in [\![D]\!]$; see [1, 22]. Under bag semantics, we have more information: for each tuple, we know the number $\#\big(\bar{a}, q(D')\big)$ of occurrences of $\bar{a}$ in $q(D')$. Thus, as D' ranges over $[\![D]\!]$, we have a range of numbers that define an interval between

$$\min(\bar{a}, q, D) = \min_{D' \in [\![D]\!]} \#\big(\bar{a}, q(D')\big) \qquad \text{and} \qquad \max(\bar{a}, q, D) = \max_{D' \in [\![D]\!]} \#\big(\bar{a}, q(D')\big)$$

Under set semantics, $\min(D, q, \bar{a}) = 1$ means that $\bar{a}$ is a certain answer to q on D, and $\max(D, q, \bar{a}) = 1$ means that $\bar{a}$ is a possible answer. On sets, these can be easily checked for positive relational algebra, but it is hard (coNP-complete and NP-complete, respectively) for full relational algebra [2]. This tells us that, in terms of relational algebra operations, selection, projection, Cartesian product and union are easy, but difference makes things hard. Our goal is to paint a similar picture for bags. The problems that we face are:

1. there are more operations that are included in bag relational algebra, and we have less understanding of them;
2. even for basic operations, there is very little knowledge of the complexity of answering queries over incomplete data.

We now explain these points in more detail.

Bag algebra fragments

Under set semantics, we have well understood fragments of relational algebra: SPC (select-project-Cartesian product) queries, positive relational algebra RA^+ that adds union, and full relational algebra RA that adds difference. Moreover, intersection is expressible as the natural join of two relations over the same attributes. Under bag semantics, however, the situation is different:

- SPC queries follow their set-theoretic analogs except that they keep duplicates.
- For union, there are several options: either the additive union, that adds up multiplicities (and corresponds to `UNION ALL` in SQL), or the set theoretic union (SQL's `UNION`), or the *max-union* which takes the maximum number of occurrences of a tuple.
- There are two standard notions of intersection: the set-theoretic one (SQL's `INTERSECT`), or the one taking the minimum number of occurrences of a tuple (SQL's `INTERSECT ALL`). Neither of them is the join of two relations any more.

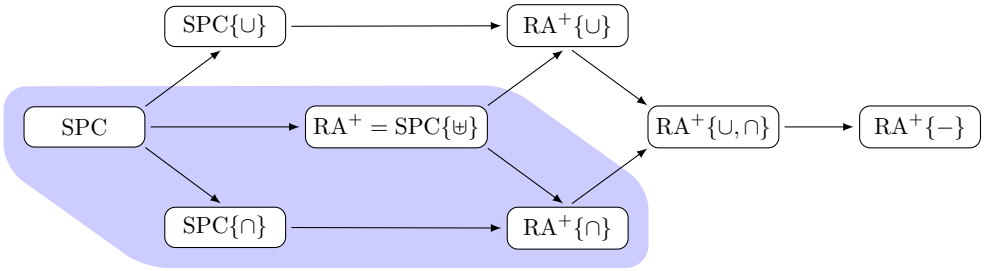

Figure 1 Summary of the results. An edge indicates a more expressive fragment. Adding duplicate elimination makes every fragment more expressive, and incomparable with $\mathrm{RA}^+\{-\}$. Extending $\mathrm{RA}^+\{-\}$ with duplicate elimination results in a fragment that has the expressive power of full RA. The shaded area includes the fragments for which computing the minimum occurrences of certain answers is tractable, while this is intractable whenever duplicate elimination is added. Computing the maximum number of occurrences is intractable for all fragments.

- There are also two versions of the difference operator, corresponding to SQL's `EXCEPT` and `EXCEPT ALL`: the former removes duplicates, and the latter keeps them.
- Finally, there is the duplicate elimination operation, that corresponds to SQL's `DISTINCT`.

The language RA of bag relational algebra then has the following operations [4, 18, 26]:

- multiplicity-preserving versions of *selection*, *projection* and *Cartesian product*, which form the class of SPC queries;
- *additive union* $\uplus$ that adds up multiplicities of tuples; together with SPC queries it forms the *positive relational algebra* RA^+;
- *max-union* $\cup$ that keeps the maximum number of occurrences of a tuple;
- *min-intersection* $\cap$ that keeps the minimum number of occurrences of a tuple;
- *difference* $-$ that subtracts the number of occurrences of a tuple up to zero, i.e., $\#(\bar{a}, R - R') = \max\big(\#(\bar{a}, R) - \#(\bar{a}, R'), 0\big)$;
- *duplicate elimination* ε that turns a bag into a set.

To understand how query answering behaves in these fragments, we first need to understand their relative expressiveness. It *might* appear that these questions have already been answered in [17, 26]. However, this was done in the context of nested relations, and the results used the power of nesting in an essential way. For the usual bag algebra with non-nested relations, as implemented in all DBMSs, these basic results are surprisingly lacking. Thus, as our first task, we shall produce a full picture of the expressiveness of bag relational algebra fragments (which will indeed be different from the known results in the nested case).

Incomplete information and bags

There is a much bigger variety of relational algebra fragments for bags, but little is known about finding $\min(\bar{a}, q, D)$ and $\max(\bar{a}, q, D)$ for queries in those fragments. We know that min is easy to compute for RA^+ queries and that for full RA the problem is computationally hard: checking whether $\min(\bar{a}, q, D) \geq n$ is NP-complete [2, 14]. Checking whether $\max(\bar{a}, q, D) \geq n$ is NP-complete even for SPC queries [14]. The complexity of actually computing min and max (or, in terms of a decision problem, checking whether $\min(D, q, D) = n$, and likewise for max) is still open.

Outline of the results

Our main results are summarized in Figure 1.

Expressiveness We characterize the relative expressive power of RA fragments, as shown in the diagram. Furthermore, adding duplicate elimination to a fragment that does not have it results in a language that is strictly more expressive (than the original fragment), and incomparable with $\mathrm{RA}^+\{-\}$. The relative expressiveness of bag operations is indeed different from what was known in the nested relational case [17, 18, 26]. For example, over nested relations, adding min-intersection to the analog of RA^+ suffices to express max-union, but in the usual relational algebra over bags these two operations are incomparable in their expressiveness.

Complexity of min For fragments in the shaded area, computing $\min(D, q, \bar{a})$ is tractable, and it can be done by evaluating the query naively on the incomplete database. For all fragments outside the shaded area, and all fragments with duplicate elimination (from $\mathrm{SPC}\{\varepsilon\}$ to the full RA), the complexity is intractable: checking whether $\min(\bar{a}, q, D)\ \theta\ n$ is NP-complete when θ is $\leq$, coNP-complete when θ is $\geq$, and DP-complete when θ is $=$. Recall that DP is the class of problems that are the intersection of an NP problem and a coNP problem [29].

Complexity of max For all the fragments, inside and outside the shaded area, and with or without duplicate elimination, computing max is intractable: checking $\max(D, q, \bar{a})\ \theta\ n$ is NP-complete when θ is $\geq$, coNP-complete when θ is $\leq$, and DP-complete when θ is $=$.

Organization

Bag relational algebra is defined in Section 2, and the relative expressive power of its fragments is studied in Section 3. Query answering over bags with nulls is discussed in Section 4, and its complexity is classified in Section 5. Concluding remarks are given in Section 6.

2 Bag Relational Algebra

e now describe the standard operations of bag relational algebra and provide their semantics [4, 17, 18, 26]. A bag is a collection of elements with associated multiplicities (numbers of occurrences); if an element b occurs n times in a bag B, we write $\#(b, B) = n$. If $\#(b, B) = 0$, it means that b does not occur in B. Sets are just a special case when $\#(b, B) \in \{0, 1\}$.

In a database D, each k-ary relation name R of the schema is associated with a bag R^D of k-tuples; we will omit the superscript D whenever it is clear from the context. We assume that the attributes of a k-ary relation are $1, \ldots, k$, i.e., we adopt the unnamed perspective [1].

Syntax

The syntax of relational algebra (RA) expressions is defined as follows:

$$
\begin{array}{rcll}
e, e' & ::= & R & \text{(base relations)} \\
& | & \sigma_{i=j}(e) & \text{(selection)} \\
& | & \pi_{\alpha}(e) & \text{(projection)} \\
& | & e \times e' & \text{(Cartesian product)} \\
& | & e \uplus e' & \text{(additive union)} \\
& | & e \cup e' & \text{(max-union)}
\end{array}
$$

$$
\begin{array}{lll}
\mid & e \cap e' & \text{(intersection)} \\
\mid & e - e' & \text{(difference)} \\
\mid & \varepsilon(e) & \text{(duplicate elimination)}
\end{array}
$$

where i and j in $\sigma_{i=j}(e)$ are positive integers, and α in $\pi_\alpha(e)$ is a possibly empty tuple of positive integers.

The *arity* of RA expressions is defined as follows: for base relations, it is given by the schema; for $\sigma_{i=j}(e)$ and $\varepsilon(e)$, it is the arity of e; for $\pi_\alpha(e)$, it is the arity of α; for $e \times e'$, it is the sum of the arities of e and e'; for $e \star e'$ with $\star \in \{\cup, \uplus, \cap, -\}$, it is the arity of e.

We then say that an RA expression is well-formed w.r.t. a schema if: it mentions only relation names from the schema; i and j in $\sigma_{i=j}(e)$ are less than or equal to the arity of e; the elements of α in $\pi_\alpha(e)$ are less than or equal to the arity of e; the expressions e and e' in $e \star e'$, with $\star \in \{\cup, \uplus, \cap, -\}$, have the same arity. In the rest of the paper, we implicitly assume that we are always working with well-formed RA expressions.

Semantics

We give the semantics of (well-formed) RA expressions e by inductively defining the quantity $\#(\bar{a}, e, D)$, which is the number of occurrences of a tuple $\bar{a}$ (of appropriate arity) in the result of applying e to a database D. This is done as follows:

$$
\begin{aligned}
\#(\bar{a},\, R,\, D) &= \#\big(\bar{a}, R^D\big) \\
\#(\bar{a},\, \sigma_{i=j}(e),\, D) &= \begin{cases} \#(\bar{a}, e, D) & \text{if } \bar{a}.i = \bar{a}.j \\ 0 & \text{otherwise} \end{cases} \\
\#(\bar{a},\, \pi_\alpha(e),\, D) &= \sum_{\bar{a}' \colon \pi_\alpha(\bar{a}') = \bar{a}} \#(\bar{a}', e, D) \\
\#\big(\bar{a}\bar{a}',\, e \times e',\, D\big) &= \#\big(\bar{a}, e, D\big) \cdot \#\big(\bar{a}', e', D\big) \\
\#(\bar{a},\, e \uplus e',\, D) &= \#(\bar{a}, e, D) + \#(\bar{a}, e', D) \\
\#(\bar{a},\, e \cup e',\, D) &= \max\big\{\, \#(\bar{a}, e, D),\, \#(\bar{a}, e', D) \,\big\} \\
\#(\bar{a},\, e \cap e',\, D) &= \min\big\{\, \#(\bar{a}, e, D),\, \#(\bar{a}, e', D) \,\big\} \\
\#(\bar{a},\, e - e',\, D) &= \max\big\{\, \#(\bar{a}, e, D) - \#(\bar{a}, e', D),\, 0 \,\big\} \\
\#(\bar{a},\, \varepsilon(e),\, D) &= \min\big\{\, \#(\bar{a}, e, D),\, 1 \,\big\}
\end{aligned}
$$

where $\bar{a}.i$ denotes the i-th element of $\bar{a}$, $\pi_{i_1,\ldots,i_n}(\bar{a})$ is the tuple $(\bar{a}.i_1, \ldots, \bar{a}.i_n)$, and the tuples $\bar{a}$ and $\bar{a}'$ in the rule for $e \times e'$ have the same arity as e and e', respectively.

Then, for an expression e and a database D, we define $e(D)$ as the bag of tuples $\bar{a}$ of the same arity as e so that $\#\big(\bar{a}, e(D)\big) = \#(\bar{a}, e, D)$.

Fragments

The two main fragments of RA we consider are SPC, consisting of selection (σ), projection (π) and Cartesian product ($\times$), and RA^+, which is SPC extended with additive union ($\uplus$). Given a fragment $\mathcal{L}$ of RA, we write $\mathcal{L}\{\mathsf{op}_1, \ldots, \mathsf{op}_n\}$ to denote the fragment obtained by adding the RA operations $\mathsf{op}_1, \ldots, \mathsf{op}_n$ to $\mathcal{L}$. Thus, for instance, RA^+ is $\mathrm{SPC}\{\uplus\}$.

A *query* is a mapping q from databases to bags of tuples. We always assume that queries are generic, that is, invariant under permutations of the domain [1]. A query q is *expressible* in a fragment $\mathcal{L}$ of RA if there is an expression e in that fragment so that $e(D) = q(D)$ for every database D.

Then, given two fragments $\mathcal{L}$ and $\mathcal{L}'$, we say that $\mathcal{L}'$ is *at least as expressive* as $\mathcal{L}$, and write $\mathcal{L} \subseteq \mathcal{L}'$, if every query expressible in $\mathcal{L}$ is also expressible in $\mathcal{L}'$. We say that $\mathcal{L}'$ is *more expressive* than $\mathcal{L}$, and write $\mathcal{L} \subsetneq \mathcal{L}'$, if $\mathcal{L}'$ is at least as expressive as $\mathcal{L}$ and there is a query that is expressible in $\mathcal{L}'$ but not in $\mathcal{L}$. Notice that if $\mathcal{L}'$ has all the operations of $\mathcal{L}$, then $\mathcal{L}'$ is at least as expressive as $\mathcal{L}$.

3 Expressive Power of Bag Relational Algebra Fragments

In this section, we study the relative expressiveness of RA fragments. We present the results that justify the edges in Figure 1, along with additional results that are not explicitly captured in that diagram.

We start by showing that extending positive relational algebra with max-union or intersection results in a more expressive fragment.

▶ **Proposition 1.** $\mathrm{RA}^+ \subsetneq \mathrm{RA}^+\{\star\}$ *for* $\star \in \{\cup, \cap\}$.

Proof sketch. We only need to show that there exists a query that is expressible in $\mathrm{RA}^+\{\star\}$ but not in RA^+. To this end, consider a schema consisting of two nullary (i.e., of arity 0) relation symbols R and S, and suppose that $R \star S$ is expressible in RA^+, i.e., there exists an RA^+ expression e equivalent to $R \star S$. For a database D, let $m = |R^D|$ and $n = |S^D|$; then, $|e(D)| = f_\star(m, n)$, where $f_\cup = \max$ and $f_\cap = \min$, and it is expressible by a polynomial $p_e \in \mathbb{N}[m, n]$, because $e \in \mathrm{RA}^+$ [16]. For $n_0 \in \mathbb{N}$, define the polynomial $p \in \mathbb{N}[m]$ such that $p(m) = p_e(m, n_0)$. When $\star = \cup$, we choose $n_0 = \deg(p_e)$ to derive a contradiction of the fact that $\tilde{p}(m) = d$ is the unique polynomial of degree at most d that interpolates the $d + 1$ data points $(0, d), \ldots, (d, d)$. When $\star = \cap$, we choose $n_0 > 0$ to derive a contradiction of the fact that p is strictly increasing in the interval $[0, +\infty)$, since its coefficients are in $\mathbb{N}$. ◀

The proof of Proposition 1 also applies to show the following.

▶ **Corollary 2.** $\mathrm{SPC} \subsetneq \mathrm{SPC}\{\star\}$ *for* $\star \in \{\cup, \cap\}$. ◀

We now show that additive union increases the expressive power of $\mathrm{SPC}\{\cap\}$ and $\mathrm{SPC}\{\cup\}$.

▶ **Proposition 3.** $\mathrm{SPC}\{\star\} \subsetneq \mathrm{RA}^+\{\star\}$ *for* $\star \in \{\cup, \cap\}$.

Proof sketch. On databases with nullary relations of size 1, every $\mathrm{SPC}\{\star\}$ expression with $\star \in \{\cap, \cup\}$ can only yield a (nullary) relation of size 1, while there are RA^+ expressions (e.g., $R \uplus R$ where R is a base relation) whose results size on such databases is greater than 1. ◀

In particular, the proof of Proposition 3 also applies to show the following.

▶ **Corollary 4.** $\mathrm{SPC} \subsetneq \mathrm{RA}^+$. ◀

Next, we show that $\cup$ increases the expressive power of $\mathrm{RA}^+\{\cap\}$, and $\cap$ increases the expressive power of $\mathrm{RA}^+\{\cup\}$. In turn, this implies that $\cup$ and $\cap$ are incomparable operations.

▶ **Proposition 5.** $\mathrm{RA}^+\{\star\} \subsetneq \mathrm{RA}^+\{\cup, \cap\}$ *for* $\star \in \{\cup, \cap\}$.

Proof sketch.

- $\mathrm{RA}^+\{\cup\} \subsetneq \mathrm{RA}^+\{\cap,\cup\}$

  We show that, on a schema consisting of two nullary relation symbols R and S, $e = R \cap S$ is not expressible in $\mathrm{RA}^+\{\cup\}$. To this end, let e' be an $\mathrm{RA}^+\{\cup\}$ expression over R and S. Then, $|e'(D)|$ is given by a function in the variables $m = |R^D|$ and $n = |S^D|$ consisting of sum, product and max. When e' mentions R, for all databases D where $m > n$ we have that $|e'(D)| \geq m > n = |e(D)|$. When e' mentions S, for all databases D where $m < n$ we have that $|e'(D)| \geq n > m = |e(D)|$. Therefore e' cannot be equivalent to e.
- $\mathrm{RA}^+\{\cap\} \subsetneq \mathrm{RA}^+\{\cap,\cup\}$

  We show that, on a schema consisting of two nullary relation symbols R and S, $e = R \cup S$ is not expressible in $\mathrm{RA}^+\{\cap\}$. To this end, let e' be an $\mathrm{RA}^+\{\cap\}$ expression over R and S. It can be shown that e' can be equivalently rewritten to $e_1 \cap \cdots \cap e_k$ where each e_i is an RA^+ expression. Note that this applies only for nullary relations, which suffices for our purposes, but it does not hold in general. Then, for every database D with $m = |R^D|$ and $n = |S^D|$, we have that $|e(D)| = \max(m,n) = \min(p_1,\ldots,p_k) = |e'(D)|$, where $p_i \in \mathbb{N}^+[m,n]$. Now, let n_0 be an integer greater than 1; then, $\max(m,n_0) = \min(p'_1,\ldots,p'_k)$ where each p'_i is a polynomial in $\mathbb{N}^+[m]$ such that $p'_i(m) = p_i(m,n_0)$. One of these polynomials must be the straight line $p(m) = m$, which leads to a contradiction for $m < n_0$. ◀

▶ **Corollary 6.** *$\mathcal{L}\{\cup\}$ and $\mathcal{L}\{\cap\}$ are incomparable, for $\mathcal{L} \in \{\mathrm{SPC}, \mathrm{RA}^+\}$.* ◀

Finally, we show that with additive union and difference one can express both intersection and max-union, therefore $\mathrm{RA}^+\{-\}$ is the most expressive fragment of RA without duplicate elimination.

▶ **Proposition 7.** $\mathrm{RA}^+\{\cap,\cup\} \subsetneq \mathrm{RA}^+\{-\}$.

Proof sketch. To show that every $e \in \mathrm{RA}^+\{\cap,\cup\}$ can be expressed in $\mathrm{RA}^+\{-\}$, we proceed by induction: the base case is when e is an RA^+ expression, which is trivially in $\mathrm{RA}^+\{-\}$; in the inductive step, we use the fact that, for every $x, y \in \mathbb{N}$, $\min(x,y) = x \dot{-} (y \dot{-} x)$ and $\max(x,y) = x + (y \dot{-} x)$, where $x \dot{-} y = \max(x-y, 0)$ is the *monus* operation. That $\mathrm{RA}^+\{-\}$ is more expressive than $\mathrm{RA}^+\{\cap,\cup\}$ then follows from the fact that every query expressible in $\mathrm{RA}^+\{\cap,\cup\}$ is monotone, while in $\mathrm{RA}^+\{-\}$ one can express non-monotone queries. ◀

Then, obviously, we immediately get the following.

▶ **Corollary 8.** *$\mathrm{RA}^+\{-,\varepsilon\}$ and RA have the same expressive power.* ◀

We conclude this section by showing that adding duplicate elimination to a fragment that does not already have it increases its expressive power.

▶ **Proposition 9.** *$\mathcal{L} \subsetneq \mathcal{L}\{\varepsilon\}$ for every fragment $\mathcal{L}$ of RA without duplicate elimination.*

Proof sketch. On databases with non-empty nullary relations of even size, every $\mathcal{L}$ expression can only yield a (nullary) relation of even size, whereas duplication elimination allows one to obtain, from such databases, relations of size 1. ◀

4 Certain Answers under Bag Semantics

Dealing with incomplete information is a recurring topic in many different areas of logic and computer science. In database theory, the main way to deal with the lack of information is via *incomplete databases*. Intuitively, an incomplete database D is a compact representation of a possibly infinite collection $[\![D]\!]$ of complete databases, which define the *semantics* of D.

In this paper, we use incomplete databases with *marked* (or *labeled*) *nulls*. This model of incompleteness is very common in the database literature [1, 22] and naturally occurs in many different scenarios, e.g., in data exchange and integration (cf. [6, 8, 25]). In this model databases are populated by *constants* and *nulls*, coming from two disjoint and countably infinite sets denoted by Const and Null, respectively. More formally, a k-ary relation is a finite bag of k-ary tuples over $\mathsf{Const} \cup \mathsf{Null}$. A database D then maps each k-ary relation symbol R in the schema to a k-ary bag relation R^D. Given a database $D = \{R_1^D, \ldots, R_n^D\}$, we write $\mathsf{Const}(D)$ and $\mathsf{Null}(D)$ for the set of constants and nulls occurring in the R_i^Ds, respectively. The *active domain* of D is the set $\mathsf{Const}(D) \cup \mathsf{Null}(D)$, denoted by $\mathsf{adom}(D)$. We say that D is *complete* if $\mathsf{Null}(D) = \varnothing$.

The semantics $[\![D]\!]$ of a database D is defined by means of valuations. A valuation v is a map $v\colon \mathsf{Null}(D) \to \mathsf{Const}$, and the result of applying v to D is the complete database vD obtained by replacing each null $\bot \in \mathsf{Null}(D)$ with $v(\bot)$. Observe that applying v to each relation R^D in D preserves multiplicities, i.e., for each relation name R in the schema and each tuple $\bar{c} \in R^{vD}$ the following equality holds: $\#\big(\bar{c}, R^{vD}\big) = \sum_{\bar{a}\colon v\bar{a}=\bar{c}} \#\big(\bar{a}, R^D\big)$. The set $[\![D]\!]$ is defined as $[\![D]\!] = \{vD \mid v \text{ is a valuation}\}$.

When relations are sets, the standard way to answer a query q on an incomplete database D is to compute *certain answers*, i.e., tuples that are in $q(D)$ for every $D' \in [\![D]\!]$, and *possible answers*, i.e., tuples that are in $q(D)$ for some $D' \in [\![D]\!]$. When relations are bags, however, one must also take multiplicities into account. In what follows, this is done by computing the minimum and maximum number of occurrences of a tuple in the answers across all databases in $[\![D]\!]$ (cf. [14]). Let D be a database, let q be a relational algebra expression of arity n, and let $\bar{a} \in \mathsf{Const}(D)^n$ be a tuple of constants, we define $\min(\bar{a}, q, D)$ and $\max(\bar{a}, q, D)$ as follows:

$$\min(\bar{a}, q, D) = \min_{vD \in [\![D]\!]} \#\big(\bar{a}, q(vD)\big) \quad ; \quad \max(\bar{a}, q, D) = \max_{vD \in [\![D]\!]} \#\big(\bar{a}, q(vD)\big) \tag{1}$$

Intuitively, $\min(\bar{a}, q, D)$ and $\max(\bar{a}, q, D)$ are extensions of certain and possible answers to bag databases. Indeed, $\min(\bar{a}, q, D) \geq 1$ if and only if $\bar{a}$ is in $q(D')$ for every $D' \in [\![D]\!]$ (and thus it is a certain answer), and $\max(\bar{a}, q, D) \geq 1$ if $\bar{a}$ is in $q(D')$ for some $D' \in [\![D]\!]$ (and thus it is a possible answer).

Thus, from now on we assume $\min(\bar{a}, q, D)$ and $\max(\bar{a}, q, D)$ as our standard notion of query answers and study their complexity. More specifically, we will focus on *data complexity*, that is, computing $\min(\bar{a}, q, D)$ and $\max(\bar{a}, q, D)$ for a fixed query q. Depending on the type of comparison we use, several decision problems arise from the computation of min and max. These decision problems are defined as follows.

Problem:	$\mathsf{MIN}^\theta[q]$, for $\theta \in \{>, =, <\}$ and a query q of arity n.
Inputs:	an incomplete database D, a tuple $\bar{a} \in \mathsf{Const}(D)^n$, a non-negative integer k.
Question:	is $\min(\bar{a}, q, D)\ \theta\ k$?

Problem:	$\mathsf{MAX}^\theta[q]$, for $\theta \in \{>, =, <\}$ and a query q of arity n.
Inputs:	an incomplete database D, a tuple $\bar{a} \in \mathsf{Const}(D)^n$, a non-negative integer k.
Question:	is $\max(\bar{a}, q, D)\ \theta\ k$?

Whether the number k is represented in unary or binary form does not matter: the results will be the same regardless. All tractability results are shown assuming binary representation, and all matching hardness results will be proved for the case when k is represented in unary. The choice of inequalities, i.e, $<$ vs $\leq$ or $>$ vs $\geq$, is not important: since k is an integer, $\leq k$ is the same condition as $< k + 1$.

As for the case of certain and possible answers, the complexity of the above problems depends on the fragment of RA in which the query q is expressed. While for some of these fragments the problems can be proved to be intractable, there are fragments of RA for which computing $\min(\bar{a}, q, D)$ is tractable and can actually be done via *naive evaluation*. The naive evaluation of a query q of arity n on a bag database D is defined as the bag obtained by assuming that each null value in $\mathsf{Null}(D)$ is a distinct constant and evaluating q directly over D. In what follows, we will denote by $\mathsf{naive}(\bar{a}, q, D)$ the number of occurrences of a tuple $\bar{a} \in \mathsf{Const}(D)^n$ in the result of the naive evaluation of an RA query q on an incomplete database D. It is well known that $\mathsf{naive}(\bar{a}, q, D)$ can be computed in DLogSpace in data complexity [18, 26].

5 Complexity of Certain Answers

We now turn our attention to the complexity of evaluating bag relational algebra expressions on incomplete databases, that is, solving the problems $\mathsf{MIN}^\theta[q]$ and $\mathsf{MAX}^\theta[q]$ for queries in various fragments of RA. As already explained, these problems are natural bag analogs of the notions of certainty and possibility over set databases.

In Section 5.1, we first provide upper and lower bounds for full RA. When the relation θ is $<$ or $>$, the results are easily derivable from the results for the set case in [2]; we complement them with the exact complexity for equality.

Then, in Section 5.2, we focus on the problem $\mathsf{MIN}^\theta[q]$ and prove the exact tractability boundary shown in Figure 1. We start by showing that in all of the fragments up to $\mathrm{RA}^+\{\cap\}$, the value of min can be computed by naive evaluation of queries, which extends a result in [14]. We then show that outside this fragment the problem is intractable, namely NP-complete for $<$, coNP-complete for $>$, and DP-complete for $=$. In particular, we show that the problem is intractable for all fragments containing $\mathrm{SPC}\{\cup\}$, and all fragments containing $\mathrm{SPC}\{\varepsilon\}$.

Next, in Section 5.3, we look at $\mathsf{MAX}^\theta[q]$. It was shown in [14] that for $>$ the problem is NP-complete, even for very simple queries. Here, we complete the picture and settle the case for $=$, even when q is a query that simply returns a relation from the database.

Finally, in Section 5.4, we discuss what happens with more complex selection conditions in queries.

5.1 Upper and lower bounds for full RA

Before delving into the complexity of the different fragments of RA, we briefly look at the complexity of evaluating general expressions. First, observe that RA queries are *generic*, i.e., invariant under permutations of the domain. In the bag case, this is stated as follows.

▶ **Proposition 10.** *Let D and D' be* complete *databases, let ρ be a bijection between $\mathsf{adom}(D)$ and $\mathsf{adom}(D')$ such that $D' = \rho(D)$, and let $q \in \mathrm{RA}$ be a query of arity n. Then, $\#(\bar{a}, q, D) = \#(\rho\bar{a}, q, D')$ for every tuple $\bar{a} \in \mathsf{adom}(D)^n$.*

Intuitively, this tells us that we need to take into account only finitely many valuations in order to compute the values in (1). Upper bounds of NP, coNP, and DP follow straightforwardly.

▶ **Proposition 11.** *Let q be an expression in* RA*. Then:*

- $\mathsf{MIN}^<[q]$ *and* $\mathsf{MAX}^>[q]$ *are in* NP*;*
- $\mathsf{MIN}^>[q]$ *and* $\mathsf{MAX}^<[q]$ *are in* coNP*;*
- $\mathsf{MIN}^=[q]$ *and* $\mathsf{MAX}^=[q]$ *are in* DP*.*

Proof sketch. Due to Proposition 10, in order to compute min and max we need to take into account only a limited number of valuations. Like in the set case [2], for a given database D we need to take into account only those valuations whose range consists of $\mathsf{Const}(D)$ and a new distinct constant for each null in $\mathsf{Null}(D)$. Moreover, evaluating RA expressions over complete bag semantics databases is in DLogSpace in data complexity. ◀

For general RA expressions, all of these problems are complete in their respective classes.

▶ **Proposition 12.** *Let q be an expression in* RA*. Then:*

- $\mathsf{MIN}^{<}[q]$ *and* $\mathsf{MAX}^{>}[q]$ *are* NP*-hard;*
- $\mathsf{MIN}^{>}[q]$ *and* $\mathsf{MAX}^{<}[q]$ *are* coNP*-hard;*
- $\mathsf{MIN}^{=}[q]$ *and* $\mathsf{MAX}^{=}[q]$ *are* DP*-hard.*

Proof sketch. The results for $<$ and $>$ follows directly from the fact that set databases can be simulated by bag databases. Hence, the hardness results presented in [2] apply. For $=$, we can show a reductions from a very well known DP-complete problem (see, e.g., Theorem 20 for the case of $\mathsf{MIN}^{=}[q]$ and Theorem 25 for the case of $\mathsf{MAX}^{=}[q]$). ◀

Despite these high bounds, one may expect that some fragments of RA will behave better; this is what we investigate next.

5.2 Computing min

We now look at computing certain answers to bag queries, that is, values $\min(\bar{a}, q, D)$. The usual naive evaluation works for queries in the $\mathrm{RA}^{+}\{\cap\}$ fragment. Outside it, the decision version of the problem, $\mathsf{MIN}^{<}[q]$, is intractable for every fragment, and for each such fragment the complexity is exactly the same. We now prove these facts.

5.2.1 Naive evaluation for $\mathrm{RA}^{+}\{\cap\}$

While computing min is hard in the general case, there exists a large fragment of RA for which min can be computed via naive evaluation. In the set case this fragment is well known to be positive relational algebra [22] consisting of selection, projection, Cartesian product, and union. Notice that over sets the intersection of two relations is expressible by join, but over bags this is no longer the case. It turns out that the good behavior of join with respect to certain answers extends to bags, when we add intersection explicitly. Indeed, for $\mathrm{RA}^{+}\{\cap\}$, one can compute min simply by using naive evaluation.

▶ **Theorem 13.** *Let q be an* $\mathrm{RA}^{+}\{\cap\}$ *expression of arity n, let D be a database, and let* $\bar{a} \in \mathsf{Const}(D)^n$*. Then,* $\min(\bar{a}, q, D) = \mathsf{naive}(\bar{a}, q, D)$*.*

Proof sketch. One can show that for every valuation v of $\mathsf{Null}(D)$ we have

$$\sum_{\bar{b}\colon v\bar{b}=v\bar{a}} \mathsf{naive}(\bar{b}, q, D) \leq \#(v\bar{a}, q, vD)\,.$$

Since the number of occurrences of a tuple $\bar{a}$ in a relation is always a non-negative integer, we have $\mathsf{naive}(\bar{a}, q, D) \leq \#(\bar{a}, q, vD)$. The claim then follows from the fact that whenever $\mathsf{naive}(\bar{a}, q, D) = k$ there exists a valuation v of $\mathsf{Null}(D)$ such that $\#(\bar{a}, q, vD) = k$. ◀

From the fact that bag relational algebra queries are in DLogSpace with respect to data complexity [18, 26], we then immediately get the following.

▶ **Corollary 14.** *For expressions $q \in \mathrm{RA}^{+}\{\cap\}$, the problems* $\mathsf{MIN}^{<}[q]$*,* $\mathsf{MIN}^{>}[q]$ *and* $\mathsf{MIN}^{=}[q]$ *are all in* DLogSpace*.*

5.2.2 Hardness for SPC{∪} and fragments with duplicate elimination

We now show that RA$^+${∩} is the best fragment for which we can compute certain answers efficiently under bag semantics. From the diagram in Figure 1, it suffices to prove that the problem is intractable for SPC{∪}, as well as for all fragments with duplicate elimination. This is what we do here; in fact we shall see that in all of these fragments, the intractable complexity will be exactly the same, namely NP-complete, coNP-complete, or DP-complete, when θ is $<$, or $>$, or $=$.

Technical tools

To prove these results, we use reductions from two well-known decision problems: *satisfiability* (SAT) and *satisfiability-unsatisfiability* (SAT-UNSAT) of propositional formulae in conjunctive normal form (CNF). Recall that a k-CNF formula ϕ is the conjunction of clauses, where each clause is a disjunction of at most k literals, i.e., variables or their negations. In what follows, we assume k-CNF formulae with exactly k distinct literals in each clause. In our scenario, this can be done without loss of generality. We write $\mathsf{Var}(\phi)$ for the set of variables of ϕ, and $|\phi|$ for the number of clauses of ϕ. The formula ϕ is *satisfiable* if there exists an assignment for $\mathsf{Var}(\phi)$ that satisfies all the clauses of ϕ. We let k-SAT refer to the problem of checking whether a CNF formula is satisfiable; this problem is well known to be NP-complete for $k \geq 3$. Given two k-CNF formulae ϕ and ψ, SAT-UNSAT is the problem of checking whether ϕ is satisfiable while ψ is not. For two given formulae $\phi, \psi \in k$-CNF, for $k \geq 3$, checking whether ϕ is satisfiable and ψ is not satisfiable is DP-complete [29].

For our reductions, we will use two additional technical tools: an encoding of k-CNF formulae as relations and a CNF formula with specific properties. Let ϕ be a propositional k-CNF formula whose clauses are $c_1, \ldots, c_n$. Assume an injection ρ from $\mathsf{Var}(\phi)$ to Null, for each $x \in \mathsf{Var}(\phi)$, the tuples $\bar{u}^t_x$ and $\bar{u}^f_x$ are defined respectively as $(0, \rho(x))$ and $(\rho(x), 1)$.

We next associate a relation R_i with each clause c_i, for $i \in \{1, \ldots, n\}$. To this end, we assume a linear ordering over $\mathsf{Var}(\phi)$. Consider a truth assignment τ for the variables of c_i, i.e., a mapping from the k variables used in c_i to *true* and *false* that makes c_i true. For each such assignment τ, we add to the relation R_i a single occurrence of the tuple $(\bar{u}_1 \ldots \bar{u}_k)$, where each $\bar{u}_j$ is equal to $\bar{u}^t_{x_j}$ if $\tau(x_j) = \mathit{true}$ and to $\bar{u}^f_{x_j}$ if $\tau(x_j) = \mathit{false}$, and the variables are considered in the ordering we assumed on $\mathsf{Var}(\phi)$. Finally, we define R_ϕ as $\biguplus_{i=1}^n R_i$.

Observe that the number of tuples in each R_i is exactly 2^k, hence the total number of tuples in R_ϕ is at most $|\phi| \cdot 2^k$. When k is fixed, the size of R_ϕ is polynomial with respect to the size of ϕ. Relations R_ϕ enjoy the following property that will be central in our proofs.

▶ **Lemma 15.** *Let ϕ be a k-CNF formula. There exists a truth assignment of $\mathsf{Var}(\phi)$ that satisfies exactly m clauses of ϕ if and only if there exits a valuation v of $\mathsf{Null}(R_\phi)$ such that* $\mathrm{range}(v) \subseteq \{0,1\}$ *and* $\#\big((0,1)^k, vR_\phi\big) = m$.

Proof sketch. First, one can show that for each clause c_i of ϕ and for every valuation v of $\mathsf{Null}(R_i)$ there is at most one occurrence of $(0,1)^k$ in vR_i. Observe now that, for each $\bar{u} \in R_i$, $v\bar{u} = (0,1)^k$ means that either $v\big(\rho(x)\big) = 1$ for a positive literal x appearing in c_i, or $v\big(\rho(x)\big) = 0$ for a negative literal $\neg x$ appearing in c_i. If there exists a valuation v of $\mathsf{Null}(R_\phi)$ such that $\mathrm{range}(v) \subseteq \{0,1\}$ and $\#\big((0,1)^k, vR_\phi\big) = m$, the following truth assignment satisfies m clauses of ϕ: $\tau(x) = \mathit{true}$ if $v(\rho(x)) = 1$, and $\tau(x) = \mathit{false}$ otherwise. Suppose now that a truth assignment τ for $\mathsf{Var}(\phi)$ that satisfies m clauses of ϕ exists. The valuation v for $\mathsf{Null}(D)$ is such that $\#\big((0,1)^k, vR_\phi\big) = m$: $v\big(\rho(x)\big) = 1$ if $\tau(x) = \mathit{true}$, and $v\big(\rho(x)\big) = 0$ otherwise. ◀

Our second technical tool is a formula $h(f,g)$ derived from two k-CNF formulae

$$f = \bigwedge_{i=1}^{n} \left(f_i^1 \vee \cdots \vee f_i^k\right) \qquad \text{and} \qquad g = \bigwedge_{i=1}^{m} \left(g_i^1 \vee \cdots \vee g_i^k\right)$$

where each f_i^j and g_i^j is a literal. Let x and y be two propositional variables not appearing in $\mathsf{Var}(f) \cup \mathsf{Var}(g)$; then, $h(f,g)$ is the following $(k+1)$-CNF formula:

$$\bigwedge_{i=1}^{n} \left(f_i^1 \vee \cdots \vee f_i^k \vee x\right) \wedge \bigwedge_{i=1}^{m} \left(g_i^1 \vee \cdots \vee g_i^k \vee \neg x\right) \wedge \bigwedge_{i=1}^{m} \left(g_i^1 \vee \cdots \vee g_i^k \vee \neg y\right) \wedge (x) \wedge (\neg x \vee y)$$

The formula $h(f,g)$ can be used to check whether f is satisfiable while g is not, due to the following property.

▶ **Lemma 16.** *Let f and g be two k-CNF formulae. Then f is satisfiable and g is unsatisfiable if and only if the maximum number of clauses of $h(f,g)$ that can be satisfied by a single truth assignment is exactly $|h(f,g)| - 1$.*

The SPC{∪} case

We now look at the complexity of handling the max-union operator ∪. Somewhat unexpectedly, adding ∪ to SPC gives rise to a substantial increase in the complexity of computing min; recall that in the set case, adding union is harmless and preserves the property that certain answers can be found by naive evaluation. To prove this claim, we will use reductions from SAT and SAT-UNSAT.

Let ϕ be a k-CNF formula, and let D_ϕ denote the database $\{D^R, D^T\}$ where D^R is R_ϕ (i.e., the encoding of CNF formulae presented earlier) and D^T contains an occurrence of $(0,1)^k$ for each clause in ϕ, that is, $\#\big((0,1)^k, D^T\big) = |\phi|$.

For reductions, we use the query $q = \pi_\varnothing(R \cup T)$. Note that we project onto the empty set of attributes, so the only possible answers for q are either the empty bag, or bags containing one or more occurrences of the empty tuple $()$.

To prove our claims, we first need to prove the following result.

▶ **Proposition 17.** *For every k-CNF formula ϕ, the database D_ϕ has the following properties:*

1. *There exists a truth assignment for $\mathsf{Var}(\phi)$ that satisfies exactly m clauses of ϕ if and only if there exits a valuation v for $\mathsf{Null}(D_\phi)$ such that $\mathrm{range}(v) \subseteq \{0,1\}$ and $\#\big((), q, vD_\phi\big) = \big((2^k+1)\cdot|\phi|\big) - m$.*
2. *For every valuation v of $\mathsf{Null}(D_\phi)$ there exists a valuation v' for $\mathsf{Null}(D_\phi)$ such that $\mathrm{range}(v') \subseteq \{0,1\}$ and $\#\big((), q, vD_\phi\big) \geq \#\big((), q, v'D_\phi\big)$.*

Proof sketch. The first property can be proved using Lemma 15. For the second, observe that $\#\big((), q, vD_\phi\big)$ depends on the number of occurrences of $(0,1)^k$ in vR_ϕ. ◀

With these notions in place, we are now ready to present the main results of this section. We start by proving that $\mathsf{MIN}^<[q]$ is NP-hard.

▶ **Theorem 18.** *The problem $\mathsf{MIN}^<[q]$ is NP-hard for $q = \pi_\varnothing(R \cup T)$.*

Proof sketch. To prove the claim, we show a reduction from SAT. Let ϕ be a 3-CNF formula. From Proposition 17, we get $\min\big((), q, D_\phi\big) = 9|\phi| - m$, where m is the maximum number of clauses of ϕ that can be satisfied by the same truth assignment. To obtain the reduction, simply observe that satisfiability of ϕ means that all the $|\phi|$ clauses of ϕ can be satisfied with the same truth assignment. In light of this, we can conclude that ϕ is satisfiable if and only if $\min\big((), q, D_\phi\big) < 9|\phi| - |\phi| + 1 = 8|\phi| + 1$. In turn, this gives the desired reduction. ◀

The complexity of $\mathsf{MIN}^{>}[q]$ follows directly from Theorem 18, being $\mathsf{MIN}^{<}[q]$ the complement of $\mathsf{MIN}^{>}[q]$.

▶ **Corollary 19.** *There exists a query $q \in \mathrm{SPC}\{\cup\}$ such that $\mathsf{MIN}^{>}[q]$ is* coNP*-hard.*

We now turn our attention to $\mathsf{MIN}^{=}[q]$ and show that this problem is hard for $q \in \mathrm{SPC}\{\cup\}$.

▶ **Theorem 20.** *The problem $\mathsf{MIN}^{=}[q]$ is* DP*-hard for $q = \pi_{\varnothing}(R \cup T)$.*

Proof sketch. To prove the claim we discuss a reduction from SAT-UNSAT. Let f and g be two 3-CNF formulae, let $h(f,g)$ be the formula defined earlier, and let D_h be the encoding of $h(f,g)$ defined above. Using Proposition 17, one can prove that exactly $|h(f,g)| - 1$ clauses of $h(f,g)$ can be satisfied by the same truth assignment if and only if there exists a valuation v of $\mathsf{Null}(D_h)$ such that $\#\big((), q, vD_h\big) = 16|h(f,g)|+1$ and there exists no valuation v' of $\mathsf{Null}(D_h)$ such that $\#(v'D_h, q, ()) \leq 16|h(f,g)|$. By Lemma 16, this implies that f is satisfiable and g is unsatisfiable if and only if $\min\big((), q, D_h\big) = 16|h(f,g)| + 1$, proving the claim. ◀

Handling duplicate elimination

In terms of tractability, no fragment survives the addition of duplicate elimination. In this section, we will show that the decision problems for all fragments from $\mathrm{SPC}\{\varepsilon\}$ to the full relational algebra are NP-complete, coNP-complete, and DP-complete for $<$, $>$ and $=$, respectively. To prove this, we will use reductions from SAT and SAT-UNSAT.

Let ϕ be a k-CNF formula, and let $D_\phi = \{D^S\}$ be the database where the relation D^S is defined as follows. Let R_i be the relation encoding the i-th clause of ϕ, as described at the beginning of this section; then define the relation $R'_i = (\{i\} \times R_i) \uplus \{\big(i, (0,1)^k\big)\}$, where $\{i\}$ and $\{\big(i, (0,1)^k\big)\}$ are bags containing one occurrence of the tuples (i) and $(i, \underbrace{0,1,\ldots,0,1}_{2k})$, respectively. We define $D^S = \biguplus_i R'_i$. Notice that the size of D^S is at most $(2^k + 1) \cdot |\phi|$.

In the proofs we use the very simple $\mathrm{SPC}\{\varepsilon\}$ query $q = \pi_{\varnothing}\big(\varepsilon(S)\big)$, whose output is either the empty set or the set $\{()\}$ containing the empty tuple.

To prove our complexity results we need the following proposition.

▶ **Proposition 21.** *For every k-*CNF *formula ϕ, the database D_ϕ has the following properties:*

1. *There exists an assignment for $\mathsf{Var}(\phi)$ that satisfies m clauses of ϕ if and only if there exists a valuation v of $\mathsf{Null}(D_\phi)$ such that $\mathrm{range}(v) \subseteq \{0,1\}$ and $\#\big((), q, vD_\phi\big) = \big((2^k + 1) \cdot |\phi|\big) - m$.*
2. *For every valuation v of $\mathsf{Null}(D_\phi)$ there is a valuation v' of $\mathsf{Null}(D_\phi)$ such that $\mathrm{range}(v') \subseteq \{0,1\}$ and $\#\big((), q, vD_\phi\big) \geq \#\big((), q, v'D_\phi\big)$.*

Proof sketch. The first property can be proved using Lemma 15. For the second, observe that $\#\big((), q, vD_\phi\big)$ depends on the number of occurrences of $(0,1)^k$ in vR_ϕ. ◀

Using Proposition 21 we can prove the main results of this section. We start with the complexity of $\mathsf{MIN}^{<}[q]$.

▶ **Theorem 22.** *The problem $\mathsf{MIN}^{<}[q]$ is* NP*-hard for $q = \pi_{\varnothing}(\varepsilon(S))$.*

Proof sketch. To prove the claim we show a reduction from SAT. Let ϕ be a 3-CNF formula and let D_ϕ be the database encoding ϕ defined above. Using Proposition 21, we can prove that $\min\big((), q, D_\phi\big) = 9|\phi| - m$, where m is the maximum number of clauses of ϕ that can

be satisfied by the same truth assignment. To obtain the reduction, observe that if ϕ is satisfiable then $|\phi|$ of its clauses can be satisfied by the same truth assignment. In turn, this proves that ϕ is satisfiable if and only if $\min\big((), q, D_\phi\big) < 9|\phi| - |\phi| + 1 = 8|\phi| + 1$. The desired reduction follows. ◀

The complexity of $\mathsf{MIN}^{>}[q]$ follows directly from Theorem 22.

▶ **Corollary 23.** *There exists a query $q \in \mathrm{SPC}\{\varepsilon\}$ such that $\mathsf{MIN}^{>}[q]$ is* coNP*-hard.*

We now look at $\mathsf{MIN}^{=}[q]$ and prove the desired lower bound for $q \in \mathrm{SPC}\{\varepsilon\}$.

▶ **Theorem 24.** *The problem $\mathsf{MIN}^{=}[q]$ is* DP*-hard for $q = \pi_\varnothing(\varepsilon(S))$.*

Proof sketch. To prove the claim, we argue that there exists a reduction from SAT-UNSAT to $\mathsf{MIN}^{=}[q]$ when $q = \pi_\varnothing(\varepsilon(S))$. Let f and g be two 3-CNF formulae, let $h(f, g)$ be the formula defined in the previous sections, and let D_h be the encoding of $h(f, g)$ as relation defined above. Using Proposition 21, one can prove that $|h(f, g)| - 1$ clauses of $h(f, g)$ can be satisfied by the same truth assignment if and only if there exists a valuation v of $\mathsf{Null}(D_h)$ such that $\#\big((), q, vD_h\big) = 16|h(f, g)| + 1$ and there exists no valuation v' of $\mathsf{Null}(D_h)$ such that $\#\big((), q, v'D_h\big) \leq 16|h(f, g)|$. Due to Lemma 16, this implies that f is satisfiable and g is not satisfiable if and only if $\min((), q, D_h) = 16|h(f, g)| + 1$. In turn, this proves the desired reduction. ◀

5.3 Computing max: hardness for simple queries

Computing max is hard already for very simple queries: in fact, $\mathsf{MAX}^{>}[q]$ is NP-complete for queries that simply return a base relation in the database [14]. Here we complete the picture and prove that solving $\mathsf{MAX}^{=}[q]$ for the same queries is complete for DP. To this end, we will show a reduction from SAT-UNSAT to $\mathsf{MAX}^{=}[q]$.

Let ϕ be a k-CNF formula, and let D_ϕ be the database consisting of the single relation $D^R = R_\phi$, which is the encoding of ϕ defined earlier. Using this simple construction we can prove the following.

▶ **Theorem 25.** *The problem $\mathsf{MAX}^{=}[q]$ is* DP*-hard even for a query q that returns a base relation in the database.*

Proof sketch. To prove the claim, we discuss a reduction from SAT-UNSAT. Let f and g be two 3-CNF formulae, and let $h(f, g)$ be the 4-CNF formula defined earlier. Consider the database D_h defined above and assume that the query q returns R. First, one can prove that for every valuation v of $\mathsf{Null}(D_h)$ there exists a valuation v' such that $\mathrm{range}(v') \subseteq \{0, 1\}$ and $\#\big((0, 1)^k, R, vD_h\big) \leq \#\big((0, 1)^k, R, v'D_h\big)$. From this considerations and Lemma 15, one can prove that exactly $|h(f, g)| - 1$ clauses of $h(f, g)$ can be satisfied by the same truth assignment if and only if there exists a valuation v of $\mathsf{Null}(D_h)$ such that $\#\big((0, 1)^4, R, vD_h\big) = |h(f, g)| - 1$ and there exists no valuation v' of $\mathsf{Null}(D_h)$ such that $\#\big((0, 1)^4, R, v'D_h\big) \geq |h(f, g)|$. Due to Lemma 16, f is satisfiable and g is unsatisfiable if and only if $\max\big((0, 1)^k, R, D_h\big) = |h(f, g)| - 1$. In turn, this proves the desired reduction. ◀

5.4 Complex selection conditions

In our definition of relational algebra, we assumed that selection conditions are equalities $i = j$. More generally, one can define these conditions by the grammar

$$c \;:=\; (i = j) \mid c \wedge c \mid c \vee c \mid \neg c$$

allowing arbitrary Boolean combinations. We briefly describe how more complex conditions affect our results.

If we simply add conjunction, that is, selection conditions are conjunctions of equalities, nothing changes at all. This is because the cascade of selections rule, $\sigma_{c \wedge c'}(e) = \sigma_c\big(\sigma_{c'}(e)\big)$, applies under bag semantics as well.

If disjunction is added, it might appear at first that this is the same as adding max-union, because $\sigma_{c \vee c'}(e) = \sigma_c(e) \cup \sigma_{c'}(e)$. As far as finding certain answers is concerned, the fragment SPC$\{\cup\}$ is intractable, so it appears that adding disjunction to selection conditions, under bag semantics, might lead to intractability. However, this is not the case: adding disjunction to selections is weaker than adding max-union, and preserves tractability.

▶ **Proposition 26.** *Let q be a query in* RA^+ *or arity n where selection conditions are positive Boolean combinations of equalities. Then,* $\mathsf{MIN}^<[q]$ *and* $\mathsf{MIN}^>[q]$ *can be solved in* DLogSpace. *More precisely, for every database D and every tuple $\bar{a} \in \mathsf{Const}(D)^n$, the value* $\min(\bar{a}, q, D)$ *can be computed by naive evaluation:* $\min(D, q, \bar{a}) = \mathsf{naive}(D, q, \bar{a})$.

For conditions that are unrestricted Boolean combinations of equalities, the problem becomes intractable, even when these are added to the positive fragment.

▶ **Proposition 27.** *In the extension of* RA^+ *that allows arbitrary Boolean combinations of equalities as selection conditions, there exist queries q such that* $\mathsf{MIN}^<[q]$ *is* NP*-complete,* $\mathsf{MIN}^>[q]$ *is* coNP*-complete, and* $\mathsf{MIN}^=[q]$ *is* DP*-complete.*

6 Conclusions

We have provided two complete classifications: of the expressive power of fragments of bag relational algebra, and of the complexity of computing certain and possible answers in those fragments. For the complexity of certain answers, we have a dichotomy: either they can be computed efficiently by naive evaluation, or their complexity is intractable, which means NP-complete, or coNP-complete, or DP-complete (depending on how the problem is turned into a decision problem).

Directions for future work are motivated by the recent work on bag semantics in data management applications where incompleteness naturally occurs, such as data exchange [20] and OBDA [28]. Notice that we have primarily concentrated on the closed-world semantics, which as of late has been actively studied in those contexts; see, e.g., [3, 5, 19, 21, 27]. Thus we believe our results could be relevant to understanding the complexity of these applications under the closed-world assumption. As another direction for future work, we would like to study the complexity of finding certain and possible answers in the fragments of bag relational algebra under the open-world assumption. The general case is of course undecidable, but the picture for the fragments studied here is not clear. Finally, we would like to use our results as the starting point for the study of answering queries with grouping and aggregation over incomplete data, as such queries rely on bag semantics.

Acknowledgments

We thank Etienne Toussaint and the referees for their helpful comments. This work was supported by EPSRC grants M025268 and N023056.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

2 Serge Abiteboul, Paris Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. *Theoretical Computer Science*, 78(1):158–187, 1991.

3 Shqiponja Ahmetaj, Magdalena Ortiz, and Mantas Simkus. Polynomial datalog rewritings for expressive description logics with closed predicates. In *IJCAI*, pages 878–885, 2016.

4 Joseph Albert. Algebraic properties of bag data types. In *VLDB*, pages 211–219, 1991.

5 Giovanni Amendola, Nicola Leone, Marco Manna, and Pierfrancesco Veltri. Enhancing existential rules by closed-world variables. In *IJCAI*, pages 1676–1682, 2018.

6 Marcelo Arenas, Pablo Barceló, Leonid Libkin, and Filip Murlak. *Foundations of Data Exchange*. Cambridge University Press, 2014.

7 Leopoldo E. Bertossi, Georg Gottlob, and Reinhard Pichler. Datalog: Bag semantics via set semantics. *CoRR*, abs/1803.06445, 2018. `arXiv:1803.06445`.

8 Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web*, pages 218–307, 2015.

9 Peter Buneman, Shamim A. Naqvi, Val Tannen, and Limsoon Wong. Principles of programming with complex objects and collection types. *Theor. Comput. Sci.*, 149(1):3–48, 1995.

10 R. G. G. Cattell. *The Object Database Standard: ODMG-93*. Morgan Kaufmann, 1993.

11 Surajit Chaudhuri and Moshe Y. Vardi. Optimization of *Real* conjunctive queries. In *PODS*, pages 59–70, 1993.

12 Sara Cohen. Equivalence of queries combining set and bag-set semantics. In *PODS*, pages 70–79, 2006.

13 Latha S. Colby and Leonid Libkin. Tractable iteration mechanisms for bag languages. In *ICDT*, pages 461–475, 1997.

14 Marco Console, Paolo Guagliardo, and Leonid Libkin. On querying incomplete information in databases under bag semantics. In *IJCAI*, pages 993–999. ijcai.org, 2017.

15 C. J. Date and Hugh Darwen. *A Guide to the SQL Standard*. Addison-Wesley, 1996.

16 Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance semirings. In *PODS*, pages 31–40. ACM, 2007.

17 Stéphane Grumbach, Leonid Libkin, Tova Milo, and Limsoon Wong. Query languages for bags: expressive power and complexity. *SIGACT News*, 27(2):30–44, 1996.

18 Stéphane Grumbach and Tova Milo. Towards tractable algebras for bags. *J. Comput. Syst. Sci.*, 52(3):570–588, 1996.

19 André Hernich. Answering non-monotonic queries in relational data exchange. *Logical Methods in Computer Science*, 7(3), 2011.

20 André Hernich and Phokion G. Kolaitis. Foundations of information integration under bag semantics. In *LICS*, pages 1–12. IEEE Computer Society, 2017.

21 André Hernich, Leonid Libkin, and Nicole Schweikardt. Closed world data exchange. *ACM Trans. Database Syst.*, 36(2):14:1–14:40, 2011.

22 Tomasz Imielinski and Witold Lipski. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–791, 1984.

23 T. S. Jayram, Phokion G. Kolaitis, and Erik Vee. The containment problem for real conjunctive queries with inequalities. In *PODS*, pages 80–89, 2006.

24 Phokion G. Kolaitis. The query containment problem: Set semantics vs. bag semantics. In *AMW*, 2013.

25 Maurizio Lenzerini. Data integration: a theoretical perspective. In *PODS*, pages 233–246, 2002.

26 Leonid Libkin and Limsoon Wong. Query languages for bags and aggregate functions. *J. Comput. Syst. Sci.*, 55(2):241–272, 1997.

27 Carsten Lutz, Inanç Seylan, and Frank Wolter. Ontology-mediated queries with closed predicates. In *IJCAI*, pages 3120–3126, 2015.

28 Charalampos Nikolaou, Egor V. Kostylev, George Konstantinidis, Mark Kaminski, Bernardo Cuenca Grau, and Ian Horrocks. The bag semantics of ontology-based data access. In *IJCAI*, pages 1224–1230, 2017.

29 Christos H. Papadimitriou and Mihalis Yannakakis. The complexity of facets (and some facets of complexity). *J. Comput. Syst. Sci.*, 28(2):244–259, 1984.

30 Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. McGraw-Hill, 2003.